Mälardalen University Press Licentiate Theses
No. 242

A DECISION SUPPORT SYSTEM FOR INTEGRATION TEST SELECTION

Sahar Tahvili

2016

School of Innovation, Design and Engineering

ISBN 978-91-7485-282-0
ISSN 1651-9256
Printed by Arkitektkopia, Västerås, Sweden

Abstract

Software testing generally suffers from time and budget limitations. Indiscriminately executing all available test cases leads to sub-optimal exploitation of testing resources. Selecting too few test cases for execution on the other hand might leave a large number of faults undiscovered. Test case selection and prioritization techniques can lead to more efficient usage of testing resources and also early detection of faults. Test case selection addresses the problem of selecting a subset of an existing set of test cases, typically by discarding test cases that do not improve the quality of the system under test. Test case prioritization schedules test cases for execution in order to increase their effectiveness at achieving some performance goals such as: earlier fault detection, optimal allocation of testing resources and reducing overall testing effort. In practice, prioritized selection of test cases requires the evaluation of different test case criteria. Therefore this problem can be formulated as a multi-criteria decision making problem. As the number of decision criteria grows, application of a systematic decision making solution becomes a necessity. In this thesis, we propose a tool-supported framework using a decision support system, for prioritizing and selecting integration test cases in embedded system development. This framework provides a complete loop for selecting the best candidate test case for execution based on a finite set of criteria. The results of multiple case studies, done on a train control management subsystem, from Bombardier Transportation AB in Sweden, demonstrate how our approach helps to select test cases in a systematic way. This can lead to early detection of faults while respecting various criteria. Also, we have evaluated a customized return on investment metric to quantify the economic benefits in optimizing system integration testing using our framework.

Sammanfattning

Programvarutestning lider generellt av tid- och budgetbegränsningar. Att urskillningslöst utföra alla tillgängliga testfall leder till bristfälligt utnyttjande av provningsresurser. Att välja för få testfall för exekvering kan å andra sidan lämna ett stort antal fel oupptäckta. Prioritering av testfall och urvalsmetoder kan leda till tidigt upptäckande av fel och kan också möjliggöra en mer effektiv användning av provningsresurser. Testfallsval tar upp problemet med att välja en del av en befintlig uppsättning testfall, vanligen genom att förkasta testfall som inte förbättrar kvaliteten på programvaran som testas. Testfallsprioritering schemalägger testfall för exekvering för att öka deras effektivitet att uppnå givna prestatandamål, såsom: tidigare upptäckande av fel, optimal fördelning av provningsresurser och minskande av den totala mängden provning. I praktiken så kräver ett prioriterat urval av testfall utvärdering av flera olika testfallskriterier. Därför kan detta problem formuleras som ett flermålsbeslutsfattande problem. Allt eftersom antalet beslutskriterier växer, blir nyttjande av en lösning för systematiskt beslutsfattande en nödvändighet. I den här avhandlingen föreslår vi ett verktygsbaserat beslutfattningssystem för att prioritera och välja integrationstestfall vid utveckling av inbyggda system. Det här systemet ger en komplett process för att välja den bästa av testfallkandidaterna för exekvering baserat på en ändlig uppsättning kriterier. Resultaten från flera fallstudier, gjorda på ett tågkontroll-delsystem, från Bombardier Transportation i Sverige, visar hur vår metod hjälper till att på ett systematiskt sätt välja testfall. Detta kan leda till ett tidigt upptäckande av fel samtidigt som olika kriterier uppfylls. Med hjälp av en anpassad avkastning på investering metrik, så visar vi vidare att vår föreslagna beslutstödsystem ger ekonomiska fördelar med att optimera provning under systemintegration.

List of Figures

List of Tables

To my parents

Acknowledgments

Many special thanks to my main supervisor, Markus Bohlin, who has been very supportive and encouraging beyond just academical work and also to my assistant supervisors Stig Larsson, Daniel Sundmark and Wasif Afzal for all their guidance, help and support. I have learned so much from you personally and professionally, working with you made me grow as a PhD student. I would also like to express gratitude towards my manager, Helena Jerregård, who has always supported me throughout the work on this thesis. SICS is a great workplace that I very much enjoy being part of.
I would also like to thank my additional co-author Mehrdad Saadatmand, working with you is a great pleasure.
Furthermore, thanks to all my colleagues at SICS Västerås: Linnéa Svenman Wiker, Zohreh Ranjbar, Björn Löfvendahl, Pasqualina Potena, Jaana Nyfjord, Joakim Fröberg, Stefan Cedergren, Anders Wikström, Petra Edoff, Martin Joborn, Helena Junegard, Susanne Timsjö, Blerim Emruli, Daniel Flemström, Kristian Sandström, Tomas Olsson, Thomas Nessen, Niclas Ericsson, Ali Balador, Anders OE Johansson.
A special thanks to Ola Sellin, Kjell Bystedt, Anders Skytt, Johan Zetterqvist, Mahdi Sarabi and the testing team at Bombardier Transportation, Västerås, Sweden.
My deepest gratitudes to my family and my friends: Neda, Shahab, Lotta, Jonas, Razieh, Iraj and Leo who have always been there for me no matter what. Without them I could have never reached this far.
The work presented in this Licentiate thesis has been funded by SICS Swedish ICT, Vinnova grant 2014-03397 through the IMPRINT project and also the Swedish Knowledge Foundation (KK stiftelsen) through the ITS-EASY program at Mälardalen University.

Sahar Tahvili
Västerås, October 2016

List of Publications

Papers Included in the Licentiate Thesis [1] [2]

Paper A *Multi-Criteria Test Case Prioritization Using Fuzzy Analytic Hierarchy Process*. S. Tahvili, M. Saadatmand and M. Bohlin. The 10th International Conference on Software Engineering Advances (ICSEA 2015), Spain, November, 2015.

Paper B *Dynamic Test Selection and Redundancy Avoidance Based on Test Case Dependencies*. S. Tahvili, M. Saadatmand, S. Larsson, W. Afzal, M. Bohlin and D. Sundmark. The 11th Workshop on Testing: Academia-Industry Collaboration, Practice and Research Techniques (TAIC PART 2016), USA, April, 2016.

Paper C *Towards Earlier Fault Detection by Value-Driven Prioritization of Test Cases Using Fuzzy TOPSIS*. S. Tahvili, W. Afzal, M. Saadatmand, M. Bohlin, D. Sundmark, and S. Larsson. The 13th International Conference on Information Technology: New Generations (ITNG 2016), USA, April, 2016.

Paper D *Cost-Benefit Analysis of Using Dependency Knowledge at Integration Testing*. S. Tahvili, M. Bohlin, M. Saadatmand, S. Larsson, W. Afzal and D. Sundmark. The 17th International Conference on Product-Focused Software Process Improvement (PROFES 2016), Norway, November, 2016, Accepted for publication.

[1] A licentiate degree is a Swedish graduate degree halfway between M.Sc. and Ph.D.
[2] The included articles have been reformatted to comply with the licentiate layout.

Additional Papers, Not Included in the Licentiate Thesis

Journals

- *On the global solution of a fuzzy linear system*. T. Allahviranloo, A. Skuka and S. Tahvili. Journal of Fuzzy Set Valued Analysis (ISPACS), Volume 2014.

Conferences

- *An Online Decision Support Framework for Integration Test Selection and Prioritization (Doctoral Symposium)*. S. Tahvili. The International Symposium on Software Testing and Analysis (ISSTA 2016), Germany, July, 2016.

- *A Fuzzy Decision Support Approach for Model-Based Tradeoff Analysis of Non-Functional Requirements*. M. Saadatmand and S. Tahvili. The 12th International Conference on Information Technology : New Generations (ITNG 2012), USA, April, 2015.

- *Solving complex maintenance planning optimization problems using stoc hastic simulation and multi-criteria fuzzy decision making*. S. Tahvili, S. Silvestrov, J. Österberg and J. Biteus. The 10th International Conference on Mathematical Problems in Engineering, Aerospace and Sciences (ASMDA 2015), Norway, June, 2014.

Other

- *Test Case Prioritization Using Multi Criteria Decision Making Methods*. S. Tahvili and M. Bohlin. Danish Society for Operations Research (OrBit), Volume 26, 2016.

Contents

I

Thesis

Chapter 1

Introduction

In a typical software development process, certain basic activities are required for the successful execution of the project. An example of such activities is shown in Figure 1.1 as V-model of software development life cycle.

In this context, it is useful to know the specific roles played by verification and validation (V&V) activities [3]. Software testing is an important part of such activities and should be started as soon as possible. Additionally, software testing should be carried out effectively to improve the quality of the product [4].

Software V&V consists of two distinct sets of activities. Verification consists of a set of activities that checks the correct implementation of a specific function, while Validation is a set of activities that checks that the software satisfies the customer requirements. The IEEE Guide for Software Verification and Validation plans [5] precisely describes this as: "a V&V effort strives to ensure that quality is built into the software and that the software satisfies user requirements". Boehm [6] presents another way to state the distinction between software V&V:

Verification: "Are we building the product right?"
Validation: "Are we building the right product?"

1.1 Software Testing

Software testing plays an obligatory role in the software development life-cycle to recognize the difficulties in the process very well [4]. According IEEE in-

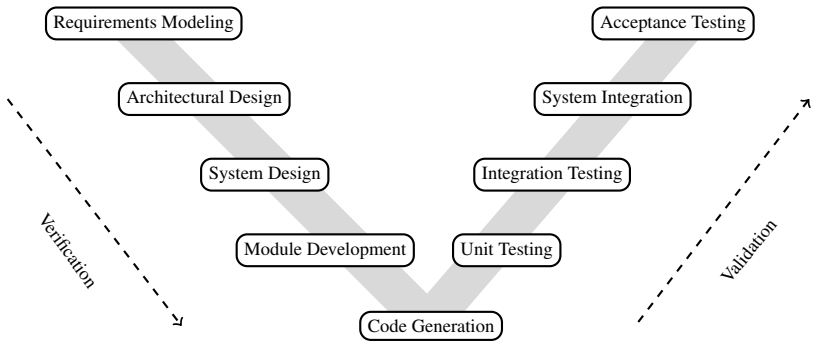

Figure 1.1: The V Model for the software development life cycle.

ternational standard (ISO/IEC/IEEE 29119-1) formulation [7],

Definition 1.1. Software testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item.

Software testing is an activity that should be done throughout the whole development process [8]. Testing generally suffers from time and budget limitations. An additional level of complexity is added when testing for integration of subsystems due to inherent functional dependencies and various interactions between integrating subsystems. Therefore, improving the testing process, especially at integration level, is beneficial from both product quality and economic perspectives.

Towards this goal, application of more efficient testing techniques and tools as well as automation of different steps of the testing process (e.g., test case generation, test execution, report generation and analysis, root cause analysis, etc.) can be considered. ISO/IEC/IEEE 29119-1 defines a test case as [7],

Definition 1.2. Set of test case preconditions, inputs (including actions, where applicable), and expected results, developed to drive the execution of a test item to meet test objectives, including correct implementation, error identification, checking quality, and other valued information.

A typical test case consists of test case description, expected test result, test step, test case ID, related requirements, etc. Table 1.1 represents an example of a manual test case for a safety critical system at Bombardier Transportation (BT). The number of test cases that are required for testing a system depends on

several factors, including the size of the system under test and its complexity. However, the decision of what test cases to select for execution and the order in which they are executed can also play an important role in efficient use of allocated testing resources.

Test case name				
Undue fourth stage				
Test case ID	Test level(s)	Pass/Fail	Comments	
SwTS-DBC-IVVP-0538 (v.1)	Sw/Hw Integration	Pass		
Test configuration				
TCMS baseline: TCMS 1.2.3.0 Test rig: VCS Release 1.16.5 VCS Platform 3.24.0				
Requirement(s)				
SRS-DBC-REQ-1306 (v.1) the actual speed is above 35km/h				
Initial State				
Initial state: Ready to drive DM1 cab active Ready to drive				
Step	Action	Reaction	Pass/fail	Comments
1	Set the train at speed 36 km/h	Check that on IDU no event regarding "undue fourth stage" is shown	Pass	
2	Lock and set from VCS Brake IO panel the signal "Relay Brake 4th stage" for DM1	Check that on IDU an event regarding "undue fourth stage" is shown	Pass	As maintainer
3	Repeat for all cars	Check that: -the steps above are fulfilled	Pass	
Clean-up				
Clean-up				

Table 1.1: A test case example from the safety-critical train control management system at Bombardier Transportation

A set of test cases which are required to test a software program are typically referred to as a test suite. A test suite often contains detailed instructions or goals for each collection of test cases and information on the system configuration to be used during testing [9].

1.2 Test case selection and prioritization

Prioritization, selection and minimization of test cases are well-known problems in the area of software testing. The test case selection problem is defined by Yoo and Harman [10] as,

Definition 1.3. Given: The program, P, the modified version P' of P, and a test suite, T.

Problem: Find an as small as feasible faults-revealing subset T' of T, with which to test P'.

In other words, test case selection strives to identify a subset of test cases that are relevant to some set of recent changes [10].

Test case prioritization deals with ordering test cases for execution. Several goals such as earlier fault detection and reducing testing effort can be achieved by prioritizing test cases. The prioritization problem is defined as follows by Yoo and Harman [10]:

Definition 1.4. Given: A test suite, T, the set of permutations of T, PT, and a function from PT to real numbers, $f : PT \rightarrow \mathbb{R}$.
Problem: To find $T' \in PT$ that maximizes f.

We recognized the process of selecting and prioritizing test cases as a multi-criteria decision making problem, where more than one criterion should be considered per time.

Multi-criteria decision making (MCDM) is a sub-discipline of operations research that explicitly considers several criteria simultaneously in a formally defined decision-making process [11]. MCDM techniques are based on criteria and alternatives, where a set of identified criteria are impacting a finite set of alternatives. We define the required test cases for testing a system under test, as a set of our alternatives. The process of identifying the critical criteria is a unique process and it depends on the problem and environment. The following set of criteria has been identified by us in a cooperation of testing experts [12] in industry:

- Requirement coverage
- Fault detection probability
- Time efficiency
- Cost efficiency
- Dependency

It should also be noted that the proposed approach in this thesis is not, however, limited to any particular set of test case properties as decision making criteria. In different systems and contexts, users can have their own set of key test case properties based on which prioritization is performed.

1.3 Decision Support Systems

The main objective of this thesis is proposing a decision support system which can help testers, dynamically prioritize and select test cases for execution at integration testing. Power et al. [13] have defined a decision support system as,

Definition 1.5. Decision support systems (DSS) are a class of computerized information system that support decision-making activities. Decision support systems are designed artifacts that have specific functionality.

Moreover, a properly designed DSS is an interactive software-based system intended to help decision makers compile useful information from raw data, documents, personal knowledge, and-or business models to identify and solve problems and make decisions [13]. Further, a dynamic decision making approach contains the following steps proposed by Bertsimas and Freund [1], mirrored in Figure 1.2.

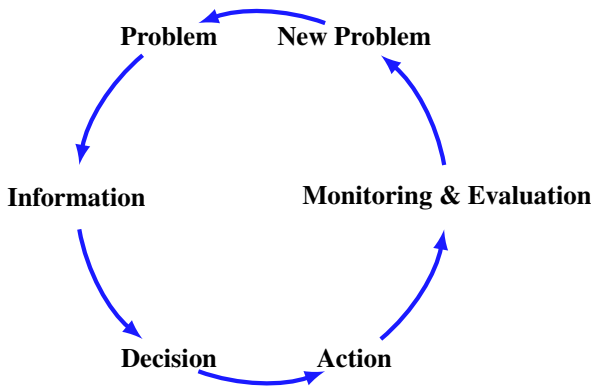

Figure 1.2: The cycle of a dynamic decision making process proposed by Bertsimas and Freund [1].

As Figure 1.2 represents, to have a dynamic process in decision making, we need to monitor, evaluate and communicate our decisions continuously. Additionally, the decisions must be quick, simple and efficient. By applying this model in our research, ordering test cases for execution (our initial problem) has been converted to multi-criteria test case selection and prioritization problem (a new problem). Through the process of gathering information and conducting research, we proposed a set of multi-criteria decision making techniques for selecting and prioritizing test cases. Also, the proposed techniques

have been applied on industrial case studies at BT. The proposed DSS process was evaluated by introducing and utilizing indicators such as fault failure rate[1] and return on investment [2].

1.4 Problem Formulation and Motivation

The lack of a dynamic approach to decision making might lead to a non-optimal usage of resources. Nowadays, the real world decision making problems are multiple criteria, complex, large scale and generally consist of uncertainty and vagueness. Most of the proposed techniques for ordering test cases are offline, meaning that the order is decided before execution, while the current execution results do not play a part in prioritizing or selecting test cases to execute. Furthermore, only few of these techniques are multi-objective whereby a reasonable trade-off is reached among multiple, potentially competing, criteria. The number of test cases that are required for testing a system depends on several factors, including the size of the system under test and its complexity. Executing a large number of test cases can be expensive in terms of effort and wall-clock time. Moreover, selecting too few test cases for execution might leave a large number of faults undiscovered. The mentioned limiting factors (allocated budget and time constraints) emphasize the importance of test case prioritization in order to identify test cases that enable earlier detection of faults while respecting such constraints. While this has been the target of test selection and prioritization research for a long time, it is surprising how only few approaches actually take into account the specifics of integration testing, such as dependency information between test cases. However exploiting dependencies in test cases have recently received much attention (see e.g., [16, 17]) but not for test cases written in natural language, which is the only available format of test cases in our context. Furthermore, little research has been done in the context of embedded system development in real, industrial context, where integration of subsystems is one of the most difficult and fault-prone task. Lastly, managing the complexity of integration testing requires decision support for test professionals as well as trading between multiple criteria; incorporating such aspects in a tool or a system is lacking in current research.

[1] The fault failure rate is an indicator to show that the hidden faults in a system under test can be detected earlier [14].

[2] Return on investment is a performance measure used to evaluate the efficiency of an investment, which is a common way of considering profits in relation to capital invested [15].

The principal objective of the thesis is proposing a systematic, multi-criteria decision making approach for integration test selection and prioritization in the context of embedded system development, exemplified by industrial case studies at Bombardier Transportation Sweden AB.

This thesis is conducted within the IMPRINT (Innovative Model-Based Product Integration Testing) project, funded by Vinnova [18], and KKS funded ITS-EASY industrial research school. IMPRINT is driven by the needs of its industrial partners (Bombardier Transportation, Volvo Construction Equipment, ABB HVDC). The studies undertaken in this thesis contribute to the overall goal of IMPRINT, which is to make product integration testing more efficient and effective.

1.5 Related Work

Use of various techniques for optimizing the software testing process has recently received much attention. Muthusamy and Seetharaman [19] use a weight factor approach for for prioritizing the test cases in regression testing. The test case prioritization approach is based on a weighted priority value algorithm.

Badhera et al. [20] presented a method for selecting a set of test cases for execution, where fewer lines of code need to be executed. In other words, the modified lines of code with minimum number of test cases are most favorable for execution.

Kaur and Goyal [21] propose a new version of a genetic algorithm for test suite prioritizing in the level of regression testing. The proposed algorithm by the authors is fully automated and prioritizes test cases within a time constrained environment on the basis of entire fault coverage.

The central research paper on test case selection and prioritization is a survey paper by Yoo and Harman [10]. This survey paper presents information regarding up to various methods for classification of test cases test, suite minimization, test case selection and test case prioritization. It also provided additional insights for meta-empirical study techniques, integer programming approach and multi-objective regression testing.

Haidry and Miller [22] proposed a new approach for test suite prioritization. The proposed approach is based on dependency between test cases in regression testing. The concept of dependency in this level has been divided into two major groups: open and close dependency. The open dependency indicates a test case which should be executed before another one but not immediately before the test case. In close dependency, a test case needs to be executed im-

mediately before the other test case [22]. The authors emphasize the need to combine dependency with other types of information to improve test prioritization. Our work contributes to fill this gap whereby test case dependencies along with a number of other criteria are used to prioritize test cases.

Catal and Mishral [23] provided the results of a systematic literature review which have been performed to examine what kind of test case selection and prioritization methods have been widely used in papers. Further, a basis for the improvement of test case prioritization research and evaluation of the current trends have been proposed by the authors.

Elbaum et al. [24] proposed a method for test case prioritization in order to assess the rate of fault detection. The proposed approach is applicable for regression testing for improving the possibilities of finding and fixing bugs.

Yoon et al. [25] propose a method for test case selection by analyzing the risk objective and also estimating the requirements of risk exposure value. Further it calculates the relevant test cases and thereby determining the test case priority through the evaluated values.

Krishnamoorthi and Mary [26] also use integer linear programming for test case prioritization based on genetic algorithms, where the proposed approach consists of four traditional techniques for test case prioritization.

Raju and Uma [27] use a cluster-based technique for prioritizing test cases. In this approach, a test case with a high runtime is more favorable, where the required number of pair-wise comparisons had been significantly reduced. The authors also present a value driven approach to system level test case prioritization through prioritizing the requirements for test. The following factors have been considered for test cases prioritization: requirements volatility, fault impact, implementation complexity and fault detection.

1.6 Thesis Outline

The thesis is organized into two parts:
Part I consists of three chapters. Chapter 1 includes an introduction to the thesis and formulates the research problem. In Chapter 2, the research overview describing detailed research goals, questions and contributions is offered. In Chapter 3, we summarize the conclusion, limitations and also present suggestions for the future work.
Part II presents the technical contributions of the thesis in the form of research papers, which are organized as Chapters 4-7.

Chapter 2

Research Overview

This chapter describes the research methodology, research questions, published papers and the scientific contributions of the thesis.

2.1 Research Methodology

As stated in [28], research in software engineering often lacks an understanding as to how empirical research in conducted in their field. Depending on the type of research question, different research methods are applicable [29]. In our research, we strive to gain knowledge of the state of the art and the state of the practice. To gain an understanding of the state of the art, it is generally recommended to conduct literature reviews [30], [31]. In this research, we have carried out a literature review in the form of related work in [32], [33], [12] and [34] aimed at getting a grasp of the main published work concerning our topic of interest.

In terms of the state of the practice, we have followed the general recommendation to conduct industrial examples of realistic sizes, as suggested by [29], [35]. Also, industrial questionnaires were performed in the form of several case studies in [33], [12] and [34], to assess the state of the practice and to validate that our work can be extended to industrial practice. Based on the results of such studies, [12], [34], we propose a decision support system for selecting and prioritizing test cases dynamically at the level of integration testing.

2.2 Research Process

Gorschek et al. [2] propose a collaborative research model between industry and academia. The original model proposed in [2] has been customized by us, where some steps have been merged with other steps. For instance, we have integrated *Static validation* (step 5 in the original model presented in [2]) into validation of academia. The resulting model, shown in Figure 2.1, was used for the included papers in this thesis.

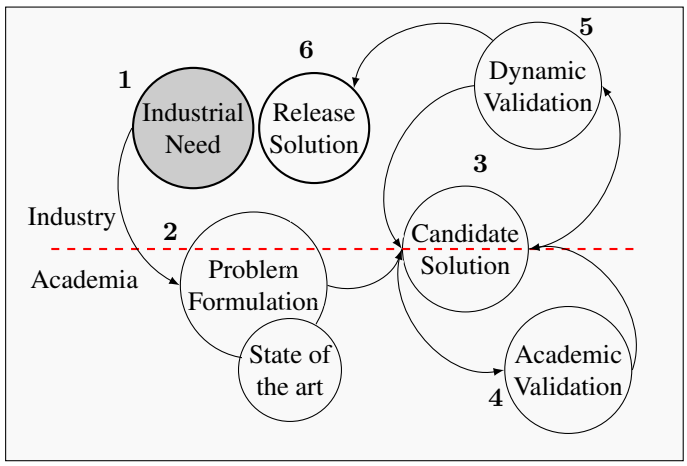

Figure 2.1: Research approach and technology transfer overview adapted from Gorschek et.al [2].

- **Step 1. Basing research agenda on industrial need**. We started our research by observing an industrial setting. The integration testing process at Bombardier transportation (BT) was selected as a potential candidate for improvement. We identified test case selection and prioritization as a real problem at BT through our process assessment and observation activities.

- **Step 2. Problem Formulation** The testing department at BT consists of various testing teams including software developers, testers, testing leaders and middle managers. The researchers (from academic partners in the research project IMPRINT [18]) have regular meetings on site. The

problem statement was formulated in close cooperation with testing experts at BT. Furthermore, interviews were utilized in this step. Moreover, the regular meetings at BT, established a common core and vocabulary of the research area and the system under test [2] between researchers and testing experts.

- **Step 3. Formulate candidate solutions** In a continuous collaboration with the testing team at BT, a set of candidate solutions for improvement of the testing process were created. The main role of BT in this part was keeping the proposed solutions compatible with their testing environment. On the other hand, the research partners were keeping track of the state of the art and applying the created solutions with a combination of new ideas [2]. Finally, with an agreement with BT, the proposed DSS was selected as the most promising solution for selection and prioritization of test cases in the integration testing level at BT.

- **Step 4. Scientific validation** Our scientific work was evaluated by international review committees of venues where we published this work ([32],[33],[12], [36] and [34]). Furthermore, an initial prototype version of the proposed DSS was implemented and evaluated in a second-cycle course by master students in the robotics program at Mälardalen University.

- **Steps 5. Dynamic Validation** This step has been performed through regular IMPRINT project meeting between all industrial and academic partners. According to the project's plan, a physical meeting should be held at the end of different phases of the project. The results of the collected case studies, prototype and experiments are presented by researchers during the meetings.

- **Step 6. Release Solution** In an agreement with BT and based on the results from academic and dynamic validation, we made the joint decision to to use the proposed DSS manually. To obtain this target, the testing experts at BT have been involved through answering our questionnaires. Also a semi-automated version has been implemented by master students and we have a plan to stepwise release the fully automated version in industry in the future.

2.3 Research Questions

As mentioned earlier, the goal of this research is to help testers to dynamically prioritize and select test cases for execution at integration testing. This is achieved by a systematic, multi-criteria decision making approach for integration testing that is empirically validated by industrial case studies at BT. We address the following research questions in this thesis:

RQ1: How does the usage of a decision support system improve the integration testing process?

RQ2: How can the dependency information between test cases be used, together with multiple other criteria, to reach a dynamic prioritization and selection of test cases at integration testing of embedded systems?

RQ3: Under what conditions and in which scenarios, is the use of the proposed DSS cost beneficial (economically motivated)?

The following published research papers cover the above research questions; also my contributions to the papers are identified.

2.4 Paper A

Multi-Criteria Test Case Prioritization Using Fuzzy Analytic Hierarchy Process, S. Tahvili, M. Saadatmand, M. Bohlin, *The 10th International Conference on Software Engineering Advances (ICSEA 2015)*, November 2015 - Spain, Published.

Abstract: One of the key challenges in software testing today is prioritizing and evaluating test cases. The decision of which test cases to design, select and execute first is of great importance, in particular considering that testing is often done late in the implementation process, and therefore needs to be done within tight resource constraints on time and budget. In practice, prioritized selection of test cases requires the evaluation of different test case criteria, and therefore, test case prioritization can be formulated as a multi-criteria decision making problem. As the number of decision criteria grows, application of a systematic decision making solution becomes a necessity. In this paper we propose an approach for prioritized selection of test cases by using the Analytic Hierarchy Process (AHP) technique. To improve the practicality of the approach in real world scenarios, we apply AHP in fuzzy an environment so that criteria values can be specified using fuzzy variables than requiring precise

quantified values. One of the advantages of the proposed decision making process it that the defined criteria with the biggest and most critical role in priority ranking of test cases is identified. We have applied our approach on an example case in which several test cases for testing non-functional requirements in a systems are defined.

My contribution. The research work presented in this work was done in collaboration with my main supervisor adjunct professor Markus Bohlin and my colleague Dr. Mehrdad Saadatmand. I am the main contributor and first author of this paper.

2.5 Paper B

Dynamic Test Selection and Redundancy Avoidance Based on Test Case Dependencies, S. Tahvili, M. Saadatmand, S. Larsson, W. Afzal, M. Bohlin and D. Sundmark, *The 11th Workshop on Testing: Academia-Industry Collaboration, Practice and Research Techniques (TAIC PART 2016)*, April 2016 - USA, Published.

Abstract: Prioritization, selection and minimization of test cases are well-known problems in software testing. Test case prioritization deals with the problem of ordering an existing set of test cases, typically with respect to the estimated likelihood of detecting faults. Test case selection addresses the problem of selecting a subset of an existing set of test cases, typically by discarding test cases that do not add any value in improving the quality of the software under test. Most existing approaches for test case prioritization and selection suffer from one or several drawbacks. For example, they to a large extent utilize static analysis of code for that purpose, making them unfit for higher levels of testing such as integration testing. Moreover, they do not exploit the possibility of dynamically changing the prioritization or selection of test cases based on the execution results of prior test cases. Such dynamic analysis allows for discarding test cases that do not need to be executed and are thus redundant. This paper proposes a generic method for prioritization and selection of test cases in integration testing that addresses the above issues. We also present the results of an industrial case study where initial evidence suggests the potential usefulness of our approach in testing a safety-critical train control management subsystem.

My contribution. The first two authors are the main contributors of the paper focusing on test case selection based on dependency, co-authors helped in study design, analysis of the data and in writing related work.

2.6 Paper C

Towards Earlier Fault Detection by Value-Driven Prioritization of Test Cases Using Fuzzy TOPSIS, S. Tahvili, W. Afzal, M. Saadatmand, M. Bohlin, D. Sundmark, and S. Larsson, *13th International Conference on Information Technology : New Generations (ITNG 2016)*, April 2016 - USA, Published.
Abstract: In industrial software testing, development projects typically set up and maintain test suites containing large numbers of test cases. Executing a large number of test cases can be expensive in terms of effort and wall-clock time. Moreover, indiscriminate execution of all available test cases typically lead to sub-optimal use of testing resources. On the other hand, selecting too few test cases for execution might leave a large number of faults undiscovered. Limiting factors such as allocated budget and time constraints for testing further emphasizes the importance of test case prioritization in order to identify test cases that enable earlier detection of faults while respecting such constraints. In this paper, we propose a multi-criteria decision making approach for prioritizing test cases in order to detect faults earlier. This is achieved by applying the TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) decision making technique combined with fuzzy principles. Our solution is based on important criteria such as fault detection probability, execution time, complexity, and other test case properties. By applying the approach on a train control management subsystem from Bombardier Transportation in Sweden, we demonstrate how it helps, in a systematic way, to identify test cases that can lead to early detection of faults while respecting various criteria.

My contribution. I am the main author of the paper, with my co-authors having academic and industrial advisory roles. I developed the models, the concept, and performed the industrial case study.

2.7 Paper D

Cost-Benefit Analysis of Using Dependency Knowledge at Integration Testing, S. Tahvili, M. Bohlin, M. Saadatmand, S. Larsson, W.Afzal and D. Sundmark, *The 17th International Conference on Product-Focused Software Process Improvement (PROFES)*, November 2016- Norway, Accepted for publication.
Abstract: In software system development, testing can take considerable time and resources, and there are numerous examples in the literature of how to improve the testing process. In particular, methods for selection and prioriti-

zation of test cases can play a critical role in efficient use of testing resources. This paper focuses on the problem of selection and ordering of integration-level test cases. Integration testing is performed to evaluate the correctness of several units in composition. Further, for reasons of both effectiveness and safety, many embedded systems are still tested manually. To this end, we propose a process, supported by an online decision support system, for ordering and selection of test cases based on the test result of previously executed test cases. To analyze the economic efficiency of such a system, a customized return on investment (ROI) metric tailored for system integration testing is introduced. Using data collected from the development process of a large-scale safety-critical embedded system, we perform Monte Carlo simulations to evaluate the expected ROI of three variants of the proposed new process. The results show that our proposed decision support system is beneficial in terms of ROI at system integration testing and thus qualifies as an important element in improving the integration testing process.

My contribution. The first two authors are the main contributors of the paper focusing on both theoretical and experimental results, with the other co-authors having academic and industrial advisory roles. The simulation part was primarily the contribution of the second author, Markus Bohlin. I developed the models, the concept, and also performed the industrial case study. Also the writing process was an iterative contribution of all authors.

2.8 Mapping of contributions to the papers

This section maps the published papers to the research questions formulated in Section 2.3. The relation between each paper and the research questions is mirrored in Table 2.1:

Research Questions	Papers
RQ1	A, B, C, D
RQ2	B, D
RQ3	D

Table 2.1: Mapping of published papers and research questions

It should be noted that there are different ways to answer RQ1, other than our proposed approach. This is still a topic of future work for us.

2.9 Scientific Contributions

The contributions of the thesis are presented in the form of research publications. The technical contributions of the licentiate thesis can be summarized as follows:

- As the first contribution of this thesis, we provided a novel multi-criteria test case prioritization method based on Fuzzy AHP[1] and TOPSIS[2]. The method is applied in a fuzzy environment to relax the need of having precise values for criteria.
- As the second contribution of this thesis, we proposed a decision support system (DSS) based on multi-criteria decision making techniques for selecting and prioritizing the best candidates (test cases) for execution. The proposed DSS consists of a dynamic and static phase which leads the testers to select the best candidate per time through observing the results of the test cases.
- As the third contribution of the present thesis, we propose and validate a parametric cost estimation model for the proposed DSS. Furthermore, we analyze the cost factors implied in the integration testing process.

[1]The analytic hierarchy process
[2]The Technique for Order of Preference by Similarity to Ideal Solution

Chapter 3

Conclusions

In this licentiate thesis, we have presented the concept of a decision support system (DSS), for prioritizing and selecting integration test cases for embedded system development. The proposed method has been applied to several industrial case studies focused on safety-critical train control subsystems. The results of the case studies show that the concept of multi-criteria test case selection and prioritization is applicable and can reduce the total required time for execution. When the testers did not follow nor consider test case dependency relations, some test cases were selected which failed due to the failure of test cases they were depending on. On the other hand, to have an earlier fault detection, test cases with higher probability of detecting faults and lower execution time, should be executed earlier. Consequently, using the proposed DSS could possibly reduce test execution efforts by avoiding test redundancies based on their dependencies and achieve a positive value for return on investment.

3.1 Future Work

There are however still some issues to resolve before the proposed DSS can be applied to a more complex testing process and we are already working on ways to overcome them. In particular, there is a need to minimize the role of testing experts for measuring the impact of various criteria on test cases, particularly for detecting dependencies. Since the manual test cases are written in natural language, we are currently investigating on finding a suitable machine learning method in natural language processing (NLP). Our hope is that by applying a

suitable NLP algorithm, we are able to predict the dependencies and also the effect of criteria on test cases automatically. Also, other forms of dependencies between test cases should be considered, for instance, state dependency, functional dependencies, etc.

3.2 Delimitations

For a fully automatic DSS for integration test selection and prioritization, there are a number of challenges that need to be resolved automatically, such as criteria identification, dependency prediction and time minimization for pair-wise comparisons.

In discussions with Bombardier Transportation (BT), four prioritization criteria were agreed upon. But there can be other applicable criteria, e.g., requirements volatility. The increase in the number of criteria is not a limitation of our proposed DSS, but it might take more time to perform the pair-wise comparisons, which is an integral part of our method (see section 1.3). We did not undertake such an analysis in this thesis. Further, the answers to the criteria were given by testing experts at BT.

This process takes time and there is a risk that testing experts might detect wrong ratings on the criteria, leading to a different prioritization of test cases. In our studies, we minimized this risk to the extent possible by involving a team of experienced test expert, software developer, troubleshooters, project leader and line managers.

We have used triangular and bell-shaped fuzzy membership function for evaluating the effect of the identified criteria on each test case in our case studies in [32], [12] and [33]. We did not compare other membership functions, e.g., L-shaped, trapezoidal membership function that might produce even more accurate solutions.

We used result dependency (fail based on fail) for creating the dependency model. If in a different context, another type of dependency such as state dependency is considered and is more relevant, the approach might not be applicable as it is, and might require some modifications. Moreover, our approach assumes that test case dependencies are identified, either manually or otherwise. We did not assess the time of identifying these dependencies but in cases where more complex dependencies exist, an automatic inference and extraction of dependencies is more feasible [37].

Bibliography

[1] D. Bertsimas and R.M. Freund. *Data, Models, and Decisions: The Fundamentals of Management Science*. MI - Management Science Series. South-Western College Publication, 2000.

[2] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. A model for technology transfer in practice. *IEEE Software*, 23(6):88–95, 2006.

[3] R. Pressman. *Software Engineering: A Practitioner's Approach*. 2010.

[4] P.A.J. Offutt, P.P. Ammann, and P.J. Offutt. *Introduction To Software Testing (South Asian Edition)*. Cambridge University Press, 2009.

[5] W. Afzal. Search-based approaches to software fault prediction and software testing, 2006. Licentiate thesis, Blekinge Institute of Technology.

[6] B.W. Boehm. *Software Engineering Economics*, pages 99–150. Springer, Germany, 2001.

[7] Software and systems engineering software testing part 1:concepts and definitions. *ISO/IEC/IEEE 29119-1:2013(E)*, pages 1–64, 2013.

[8] D. Galin. *Software Quality Assurance: From Theory to Implementation*. Alternative Etext Formats. Pearson Education Limited, 2004.

[9] H. Kahlouche, C. Viho, and M. Zendri. *An industrial experiment in automatic generation of executable test suites for a cache coherency protocol*, pages 211–226. Springer, USA, 1998.

[10] S. Yoo and M. Harman. Regression testing minimization, selection and prioritization: A survey. *Software Testing, Verification and Reliability*, 22(2):67–120, 2012.

[11] H. Weistroffer, C. Smith, and C. Narula. *Multiple Criteria Decision Support Software*, pages 989–1009. Springer, USA, 2005.

[12] S. Tahvili, M. Saadatmand, S. Larsson, W. Afzal, M. Bohlin, and D. Sundmark. Dynamic integration test selection based on test case dependencies. In *The 11th Workshop on Testing: Academia-Industry Collaboration, Practice and Research Techniques*, USA, 2016.

[13] D. J. Power, R. Sharda, and F. Burstein. *Decision support systems*. Wiley Online Library, 2015.

[14] V. Debroy and E. Wong, W. On the estimation of adequate test set size using fault failure rates. *Journal of Systems and Software*, 84(4):587–602, 2011.

[15] D. Rico. *ROI of Software Process Improvement: Metrics for Project Managers and Software Engineers*. J. Ross Publishing, Inc., 2004.

[16] J. Bell. Detecting, isolating, and enforcing dependencies among and within test cases. In *22nd International Symposium on Foundations of Software Engineering*, USA, 2014.

[17] S. Zhang, D. Jalali, J. Wuttke, K. Mucslu, W. Lam, M. Ernst, and D. Notkin. Empirically revisiting the test independence assumption. In *International Symposium on Software Testing and Analysis*, 2014.

[18] IMPRINT - Innovative Model-Based Product Integration Testing (Vinnova). `http://www.es.mdh.se/projects/382-IMPRINT___Innovative_Model_Based_Product_Integration_Testing__Vinnova_`. Accessed: 2016-09-26.

[19] T. Muthusamy and K. Seetharaman. Effectiveness of test case prioritization techniques based on regression testing. *International Journal of Software Engineering & Applications*, 5(6), 2014.

[20] U. Badhera, G. Purohit, and D. Biswas. Test case prioritization algorithm based upon modified code coverage in regression testing. 3(6), 2012.

[21] R. Abreu, P. Zoeteweij, and A. Gemund. Spectrum-based multiple fault localization. In *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, pages 88–99, USA, 2009. IEEE Computer Society.

[22] S. Haidry and T. Miller. Using dependency structures for prioritization of functional test suites. *IEEE Transactions on Software Engineering*, 39(2):258–275, 2013.

[23] C. Catal and D. Mishra. Test case prioritization: a systematic mapping study. *Software Quality Journal*, 21(3):445–478, 2013.

[24] S. Elbaum, A. Malishevsky, and G. Rothermel. Incorporating varying test costs and fault severities into test case prioritization. In *Proceedings of the 23rd International Conference on Software Engineering*, pages 329–338, USA, 2001. IEEE Computer Society.

[25] E. Yoon, M. Lee, and B. Choi. A test case prioritization through correlation of requirement and risk. *Journal of Software Engineering and Applications*, 5(10):823–835, 2012.

[26] R. Krishnamoorthi and S. Mary. Factor oriented requirement coverage based system test case prioritization of new and regression test cases. *Information and Software Technology*, (4):799–808, 2009.

[27] S. Raju and G. Uma. An efficient method to achieve effective test case prioritization in regression testing using prioritization factors. *Asian Journal of Information Technology*, 5(6), 2012.

[28] S. Easterbrook, J. Singer, M. Storey, and D. Damian. *Selecting Empirical Methods for Software Engineering Research*, pages 285–311. Springer, London, 2008.

[29] R. K. Yin. *Case Study Research: Design and Methods*. Applied Social Research Methods. SAGE Publications, 2009.

[30] B. Kitchenham and S. Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[31] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, pages 68–77, UK, 2008. British Computer Society.

[32] S. Tahvili, M. Saadatmand, and M. Bohlin. Multi-criteria test case prioritization using fuzzy analytic hierarchy process. In *The 10th International Conference on Software Engineering Advances*, Spain, 2015.

[33] S. Tahvili, W. Afzal, M. Saadatmand, M. Bohlin, D. Sundmark, and S. Larsson. Towards earlier fault detection by value-driven prioritization of test cases using fuzzy topsis. In *13th International Conference on Information Technology: New Generations*. USA, 2016.

[34] S. Tahvili, M. Bohlin, M. Saadatmand, S. Larsson, W. Afzal, and D. Sundmark. Cost-benefit analysis of using dependency knowledge at integration testing. In *The 17th International Conference on Product-focused Software Process Improvement*. Norway, 2016.

[35] P. Runeson, M. Höst, A. Rainer, and B. Regnell. *Case Study Research in Software Engineering: Guidelines and Examples*. Wiley, 2012.

[36] S. Tahvili. An online decision support framework for integration test selection and prioritization (doctoral symposium). In *The International Symposium on Software Testing and Analysis*, Germany, 2016.

[37] S. Arlt, T. Morciniec, A. Podelski, and S. Wagner. If A fails, can B still succeed? Inferring dependencies between test results in automotive system testing. In *Proceedings of the 8th IEEE International Conference on Software Testing, Verification and Validation*, Austria, 2015.